

20
5

**APPENDIX TO
DEVICE AND METHOD FOR INSPECTION
OF BAGGAGE AND OTHER OBJECTS**

ok to enter
so H

ALGO.DOC

Thursday, May 3, 1990 9:12 am

Page 1

```
#define MIN_HI 1
#define MAX_HI 2001
#define HI_INDEX 1

#define MAX_IDX 4000

/* #define TISSUE */ /* Tissue-equivalent epoxy plastic */
#define C4 /* C4 plastic explosive */
/* #define RDX */ /* RDX sheet explosive */
/* #define WG */ /* Water Gel explosive */
/* #define DYN */ /* 40# dynamite stick */

/* new way of determining low */
#define z1 .0247
#define z2 .01492
#define z3 .265
#define z4 112.6
#define z5 25.198
#define z6 .6218
#define z7 .265

/* define substance parameters */

#ifdef WG
#define c1 9.732
#define c2 6.108
#define c5 1.218
#define K0 .547
#define KL .961
#endif

#ifdef RDX
#define c1 9.732
#define c2 6.108
#define c5 1.218
#define K0 .65
#define KL .86
#endif

#ifdef C4
#define c1 9.732
#define c2 6.108
#define c5 1.218
#define K0 .6522
#define KL .87
#endif

#ifdef DYN
#define c1 570.46
#define c2 4.352
#define c5 .304
```

```
#define K0 .522
#define KL .765
#endif
```

```
#ifdef TISSUE
#define c1 3798
#define c2 3.8837
#define c5 0.993
#define K0 .655
#define KL .825
#endif
```

```
double bh(double km);
double bh(double km)
{
    return(c1*pow((km+c5),c2));
}
```

```
double Kref(double Hi,double Km,double k0);
double Kref(double Hi,double Km,double k0)
{
    return (((Hi+bh(Km))*k0*KL)/((bh(Km)*KL)+(Hi*k0)));
}
```

```
double alpha(double km);
double alpha(double km)
{
    return((z1+(z2*km)-(z2*z3))/(km*km));
}
double beta(double km);
double beta(double km)
{
    return((z4+((z6-km)*(z5/(z6-z7)))/km);
}
```

```
double newlow(double h,double km);
double newlow(double h,double km)
{
    return (h*(1/(km+(alpha(km)*(h/(h+beta(km)))))));
}
```

```
double find_Km(double hi,double Kair,double kref);
double find_Km(double hi,double Kair,double kref)
{
    /* find the Km that approximates the desired Kref given high val,k0 */
    int x,bitval;
    double lsbval,approx_kref;
```

```

lsbval = 0.8;
bitval = 0;

for (x=0;x<8;x++)
{
    bitval=(bitval<<1)|1;
    lsbval = lsbval/(double)2.0;

    approx_kref = (Kref(hi,((double).1+((double)bitval*lsbval)),Kair));
    if (approx_kref < kref)
        bitval=bitval&(0xfe) ;
}
return (((double)bitval*lsbval)+.1);
}
double findKm_Low(double hi,double low);
double findKm_Low(double hi,double low)
{
    /* find the Km that approximates the desired Low given high val,k0 */
    int x,bitval;
    double lsbval,approx_low;

    lsbval = 0.8;
    bitval = 0;

    for (x=0;x<8;x++)
    {
        bitval=(bitval<<1)|1;
        lsbval = lsbval/(double)2.0;

        approx_low = (Low(hi,((double).1+((double)bitval*lsbval))));
        if (approx_low < low)
            bitval=bitval&(0xfe) ;
    }
    return (((double)bitval*lsbval)+.1);
}

/* create the histogram */
for (hint = MIN_HI; hint < MAX_HI; hint += HI_INDEX)
{
    h = (double)hint;                                /* Get hi double value.*/

    /* Set up the header values and the KIdx */
    Hdr[HI_VALUE] = hint;
    KIdx = 0;
}

```

ALGO.DOC

Thursday, May 3, 1990 9:12 am

Page 4

```

/* Get the hi and lo kref */
hi_kref = Kref(h, 0.29, k0);
lo_kref = Kref(h, 0.8, k0);
k=lo_kref;

lastl = -100.0;
diff1 = 1000.0;
while (k<hi_kref)
{
    km=find_Km(h,k0,k);
    kr=Kref(h,km,k0);
    l=Low(h,km);
    if (((1-lastl)<diff1)&&(km>.29))
        diff1 = 1 - lastl;

    lastl = l;

    if (h>800.0)
    {
        k=k*1.04;
    } else
        k=1.01*k;    /* 1% bins */
}

/* do it again, but use diff1 to find values */
k=lo_kref;
km=find_Km(h,k0,k);
l=Low(h,km);
findl=(int)l;

/* adjust diff1 to a power of 2 */
tdiff1=0;
while ((1 << (tdiff1+1)) <= (int)diff1)
    tdiff1++;

km=find_Km_Low(h,(double)findl);
k=Kref(h,km,k0);

/* Save the minimum low and the scale factor */
Hdr[MIN_LO] = findl;
Hdr[LO_SCALE] = tdiff1;

while (k < hi_kref)
{
    km=find_Km_Low(h,(double)findl);
    k=Kref(h,km,k0);

    /* Save the necessary information into the values */
    KrefTab[KIdx] = (float)k;
}

```

```
KIdx++;

/* increment low */
findl += (1 << tdiff1);

/* increment bin count */
bincnt+=1;

}

/* Now we have the table, write out the header then the table */
Hdr[MAX_LO] = findl;
bwritten = write (fhndl, (char *)Hdr, sizeof(int)*4);
if (bwritten != (sizeof(int) * 4))
{
    printf("Error writing file\n");
    return(1);
}

/* Now write out the kref vector */
bwritten = write (fhndl, (char *)KrefTab, sizeof(float)*KIdx);
if (bwritten != (sizeof(float)*KIdx))
{
    printf("Error writing file\n");
    return(1);
}
}

/* output bin count */
printf("Total Kref bin count :%ld\n",bincnt);

/*
    Detection algorithm for above histogram
*/

/*
    * Function:
    *   DoBox
    *
    * Description:
    *   Process the box.
    *
    * Usage:
    *   DoBox (x, y)
    *
    * Inputs:
    *   x - int : the x coordinate of the candidate pixel
    *   y - int : the y coordinate of the candidate pixel
    *

```

ALGO.DOC

Thursday, May 3, 1990 9:12 am

Page 6

```

* Outputs:
*      None
*/

```

```

static void DoBox (int x, int y)
{
    int tx, ty;
    double diffH, diffL, diffK;
    double kreflo, krefhi, krefavg;
    /* int tmp; */
    double mindiff;
    Pixel *midpxl = &ScanLine[y][x];
    Pixel *pxl;

    /* Average the values for this pixel */
    AveragePixel (x, y);

    /* See if we need to do this pixel */
    if (midpxl->avghia > 2000.0)
        return;

    /*
    * Calculate the min difference value (this is calculated by using
    * twice the expected noise as the difference value).
    */
    mindiff = (10000.0/(100.0+midpxl->avghia));

    /* Now loop through the pixels doing the box */
    for (ty = y - BORDER; ty <= (y + BORDER); ty++)
    {
        /* Get the pixel */
        pxl = &ScanLine[ty][x - BORDER];

        /* Loop through the x */
        for (tx = x - BORDER; tx <= (x + BORDER); tx++, pxl++)
        {
            /* See if we need to look at this pixel (edges are no-nos) */
            if (pxl->sobel)
                continue;

            /* Average this sucker */
            AveragePixel (tx, ty);

            /* Now difference the Hi AIRS */
            diffH = midpxl->avghia - pxl->avghia;

            /* Now threshold it */
            if (diffH < mindiff)
                continue;

            /* Now difference the Lo AIRS */
            diffL = midpxl->avgloa - pxl->avgloa;

```

```
/* Now threshold it */
if ((diffL < mindiff) || (diffL == 0.0))
    continue;

kreflo=LookupKref(px1->avghia,px1->avgloa);
krefhi=LookupKref(midpx1->avghia,midpx1->avgloa);

diffK = diffH/diffL;

/* Key lookup algorithm
 * Histogram generation algorithm has been fit to this ratio
 */

krefavg=(kreflo*.8)+(.2*krefhi);

/* See if we need to histogram this point */
if ((diffK < (krefavg+(MinThreshold)))
    || (diffK > (krefavg+(MaxThreshold))))
    continue;

    midpx1->histval++;
}

if (maxhit<midpx1->histval)
    maxhit=midpx1->histval;
if(midpx1->histval > fomThresh)
    fom += (midpx1->histval - fomThresh);

if ((midpx1->histval > 0) && (midpx1->histval <200))
    histpix[midpx1->histval]++;

}
```


Date of Deposit July 1, 1993
I hereby certify that this correspondence is being facsimile
transmitted to the Patent and Trademark Office on the date
indicated, above.